How should I compute my candidates? A taxonomy and classification of diagnosis computation algorithms

Patrick Rodler

How should I compute my candidates? A taxonomy and classification of diagnosis computation algorithms*

Patrick Rodler
University of Klagenfurt
e-mail: patrick.rodler@aau.at

Abstract

This work proposes a taxonomy for diagnosis computation methods which allows their standardized assessment, classification and comparison. The aim is to *(i)* give researchers and practitioners an impression of the diverse landscape of available diagnostic techniques, *(ii)* allow them to easily retrieve the main features as well as pros and cons of the approaches, *(iii)* enable an easy and clear comparison of the techniques based on their characteristics wrt. a list of important and well-defined properties, and *(iv)* facilitate the selection of the "right" algorithm to adopt for a particular problem case, e.g., in practical diagnostic settings, for comparison in experimental evaluations, or for reuse, modification, extension, or improvement in the course of research.

1 Introduction

Diagnosis computation is one of the most integral tasks in model-based diagnosis as it allows to generate fault hypotheses, which are essential for both fault localization and repair. Due to its generality, the model-based diagnosis formalism has been used to express and tackle a wide diversity of debugging problems in application areas ranging from software [1], logic programming [2], recommender systems [3], ontologies [4] and knowledge bases [5] via hardware [6], circuits [7] and robots [8] to scheduling [9], aircrafts [10] and cars [11]. This has led to a remarkable multitude and heterogeneity of the diagnosis computation methods proposed in literature, which are often motivated by and tailored for application-specific requirements and problem cases. As a result, it is a hard task for researchers and practitioners alike to

- get an overview of existing approaches,
- identify the crucial properties of diagnostic techniques,
- assess the methods based on these properties, and
- choose the appropriate approach for a (research- or application-related) diagnostic task at hand.

With this work, we account for this by presenting a taxonomy for diagnosis computation algorithms. Specifically, we introduce and formally define a range of features which are arguably vital for a proper understanding, comparison, selection, and use of diagnostic techniques. We explain the influence of each feature on the proper selection of a diagnosis algorithm for a diagnostic task, discuss the potential impact of different feature manifestations on the performance of diagnosis algorithms, and examine relationships among the features. To demonstrate the value and application of the proposed taxonomy, we provide a multi-dimensional assessment and categorization of several important diagnostic methods in the literature.

*Supported by Austrian Science Fund, contract P-32445-N38.

2 Preliminaries

Model-based diagnosis [12] assumes a system (e.g., software, circuit, knowledge base, physical device) consisting of a set of *components* $\text{COMPS} = \{c_1, \ldots, c_n\}$ (e.g., lines of code, gates, axioms, physical constituents) which is formally described in some monotonic logical language. Beside any relevant general knowledge about the system, this *system description* SD includes a specification of the normal behavior (logical sentence $\text{BEH}(c_i)$) of all components $c_i \in \text{COMPS}$ of the form $\text{OK}(c_i) \rightarrow \text{BEH}(c_i)$. As a result, when assuming all components to be fault-free, i.e., $\text{OK}(\text{COMPS}) := \{\text{OK}(c_1), \ldots, \text{OK}(c_n)\}$, conclusions about the normal system behavior can be drawn by means of theorem provers. When the real system behavior, ascertained through *system observations* and/or *system measurements* (stated as logical sentences OBS and MEAS), is inconsistent with the system behavior predicted by SD, the normality-assumption for some of the components has to be retracted. We refer to $\langle \text{SD}, \text{COMPS}, \text{OBS}, \text{MEAS} \rangle$ as a *diagnosis problem instance (DPI)*.

For a DPI, one is interested in finding a diagnosis, i.e., a set of components whose abnormality would explain the observed incorrect system behavior. Formally, a set of components $\mathcal{D} \subseteq \text{COMPS}$ is called a *diagnosis* iff $\text{SD} \cup \text{OBS} \cup \text{MEAS} \cup \text{OK}(\text{COMPS} \setminus \mathcal{D}) \cup \text{NOK}(\mathcal{D})$ is consistent where $\text{OK}(X) := \{\text{OK}(c_i) \,|\, c_i \in X\}$ and $\text{NOK}(X) := \{\neg\text{OK}(c_i) \,|\, c_i \in X\}$. A diagnosis $\mathcal{D}$ is termed *minimal / minimum-cardinality* iff there is no diagnosis $\mathcal{D}'$ such that $\mathcal{D}' \subset \mathcal{D}$ / $|\mathcal{D}'| < |\mathcal{D}|$. For efficiency and tractability reasons, the focus in model-based diagnosis is often laid on minimal diagnoses only [13]. In particular, the minimal diagnoses are representative of all diagnoses under the *weak fault model* [14], where the system description SD contains only information about the normal behavior of the system components (and leaves the components' behavior undefined in case of failure). We restrict the study in this paper to diagnosis computation methods relying on a weak fault model. They address the following problem:

Problem 1 (Diagnosis Computation)**.**
Given: *A DPI* $\langle \text{SD}, \text{COMPS}, \text{OBS}, \text{MEAS} \rangle$*, an integer* $k \geq 1$*.*
Find: *Find* k *minimal diagnoses (satisfying a property* p*) for* $\langle \text{SD}, \text{COMPS}, \text{OBS}, \text{MEAS} \rangle$*.*

Diagnostic techniques may solve different manifestations of this problem. E.g., they might aim at computing n / all minimal diagnoses (by specifying $k := n$ / $k := \infty$), or at finding all minimum-cardinality diagnoses (by specifying $k := \infty$ and $p :=$ minimum-cardinality).

Useful for diagnosis computation and technically closely related to the concept of a diagnosis is the notion of a conflict, which is a set of components such that the assumption of all of them being fault-free is inconsistent with the current knowledge about the system. Formally, a set of components $\mathcal{C} \subseteq \text{COMPS}$ is a *conflict* iff $\text{SD} \cup \text{OBS} \cup \text{MEAS} \cup \text{OK}(\mathcal{C})$ is inconsistent. Again, we call a conflict $\mathcal{C}$ *minimal* iff there is no conflict $\mathcal{C}'$ with $\mathcal{C}' \subset \mathcal{C}$. There are two important links between diagnoses and conflicts [12]:

Hitting-set Property A (minimal) diagnosis is a (minimal) hitting set of all minimal conflicts.
(X is a *hitting set* of a collection of sets $\mathbf{S}$ iff $X \subseteq \bigcup_{S \in \mathbf{S}} S$ and $X \cap S \neq \emptyset$ for all $S \in \mathbf{S}$.)

Duality Property X is a diagnosis iff $\text{COMPS} \setminus X$ is not a conflict.

In many cases, there is a substantial number of competing diagnoses for a given DPI. The goal is then to isolate the *actual diagnosis* which pinpoints the *actually* faulty components. Basically, two strategies exist to handle multiple diagnoses, aiming at reducing or avoiding the effort for a manual inspection of the diagnoses: *(i) rank or restrict the computed diagnoses* based on some informative criterion such as maximal probability or minimal cardinality [13], or *(ii) apply sequential diagnosis techniques* to acquire additional information about the diagnosed system to gradually refine the set of diagnoses [7]. Whereas rankings or focusing techniques can be very powerful if the actual diagnosis appears (early) in the solution list, there is no guarantee that the actually faulty components will be located (efficiently).

The more sophisticated sequential diagnosis techniques, on the other hand, gather further system measurements (MEAS) to iteratively rule out spurious diagnoses. They aim at solving the following problem (with highest efficiency):

Problem 2 (Sequential Diagnosis)**.**
Given: *A DPI* $\langle \text{SD}, \text{COMPS}, \text{OBS}, \text{MEAS} \rangle$*.*
Find: *A sequence (set) of measurements, expressed as logical sentences* $m_1, \ldots, m_k$*, such that there is a single (highly probable) minimal diagnosis for* $\langle \text{SD}, \text{COMPS}, \text{OBS}, \text{MEAS} \cup \{m_1, \ldots, m_k\} \rangle$*.*

Many sequential diagnosis methods can be characterized by a recurring execution of four phases [15]: *(1)* computation of a set of (preferred, e.g., most probable) minimal diagnoses, *(2)* selection of the most informative system measurement based on these, *(3)* conduction of measurement actions (by some user or oracle, e.g., an electrical engineer if a circuit is diagnosed), and *(4)* exploitation of the measurement outcome to update the system knowledge. That is, the DPI is modified (by extending MEAS) between each two iterations of these phases. The execution stops if Problem 2 is solved, i.e., sufficient diagnostic certainty is achieved.

3 A Taxonomy for Diagnosis Algorithms

In this section, we propose a collection of pivotal features of diagnosis computation algorithms, based on which we will classify and compare some important existing techniques in Sec. 4 and Tab. 1. In the following, we assume that an algorithm A addresses (some manifestation of) the diagnosis computation problem (cf. Problem 1) and is given as input a DPI and possibly some meta information (such as component failure rates that allow to derive diagnosis probabilities [7], or algorithm-specific parameters, e.g., stop, pruning or restart criteria [16; 17]).[1] We describe each feature by giving a *definition* of its possible manifestations, a brief explanation of its *relevance* wrt. algorithm selection for a diagnostic task, a short discussion of the practical *impact* of different feature manifestations, and a comment on the *relationship* to other features. The features can be logically grouped into five categories, i.e., *Output Qualities* (Bullets 1–4), *Way of Computation* (5–6), *Sequential Diagnosis Context* (7–8), *Application Context* (9–11), and *Performance* (12), as shown in Tab. 1:

1. **Soundness:**
   Definition: A is *sound* iff it outputs only minimal diagnoses; otherwise, it is *unsound*.
   Relevance: Soundness is necessary if *(a)* no actually fault-free components should be marked as faulty in a diagnostic scenario, e.g., when inspecting or changing parts unnecessarily is costly such as in a car [18], or when a modification of correct components impacts the quality of the system such as for axioms in a knowledge base [19], or if *(b)* every solution returned by the algorithm should indeed be a possible explanation for the observed system misbehavior, e.g., to avoid the necessity of additional computations and of a potentially costly post-processing of the solutions. Apart from that, the soundness requirement is in line with the generally accepted principles of Parsimony [12] and Occam's razor [20], which postulate that from two different (fault) explanations, the simpler one is preferable.
   Impact: Forgoing the requirement of soundness can lead to a higher efficiency of diagnosis computation, as certain unsound algorithms are designed to drop soundness to the benefit of performance (e.g., [21; 16]). There are basically two forms of unsoundness for returned diagnoses, i.e., they might be *(a)* non-minimal diagnoses (intuitively: "too large" component sets; cf., e.g., [16]), or *(b)* non-diagnoses (intuitively: "too small" component sets; cf., e.g., [21]). Both cases can be handled by a suitable post-processing of the returned solutions (cf., e.g., [22]), the cost of which depends on the number of solutions that are non-(minimal) diagnoses and on their degree of unsoundness (i.e., how much "too small" or "too large" the diagnoses are).
   Relationship: Unsoundness can entail incompleteness or a violation of the best-first property, e.g., for systematic hitting set searches (e.g., [19]).
2. **Completeness:**
   Definition: If A computes a set of diagnoses, it is *all-complete* iff it outputs all minimal diagnoses given sufficient time and memory, and it is *property-complete* iff it outputs all minimal diagnoses with a particular property (e.g., minimum cardinality) given sufficient time and memory. If A computes a single diagnosis (with a

[1] Works describing diagnosis algorithms use a wide variety of notations and formalisms, which can however also be expressed by means of Reiter's general theory [12], as reviewed in Sec. 2.

particular property p, e.g., minimum-cardinality), it is *one-complete* iff it outputs a minimal diagnosis (with property p) whenever such a diagnosis exists. Otherwise, if there is any minimal diagnosis that A might fail to compute, it is *incomplete*.
Relevance: Completeness is necessary if it is crucial in a diagnostic scenario that the actual diagnosis is found with certainty, or when missing the actual diagnosis or a diagnosis with a particular property might have serious consequences, e.g., when diagnosing critical systems such as aircrafts, medical ontologies or security software. Moreover, completeness is vital for reasonable stop conditions in sequential diagnosis scenarios, e.g., if a single diagnosis remains after taking some measurements, only completeness implies that this diagnosis is the only possible minimal fault.
Impact: Forgoing the requirement of completeness allows for a higher efficiency of diagnosis computation in many cases, as incomplete algorithms are often devised with a particular focus on performance, cf., e.g., [16; 17; 21].
Relationship: If not carefully devised, incomplete algorithms will usually not be best-first.

3. **Best-First Property:**
Definition: A is *generally best-first* iff it computes and outputs diagnoses in order according to a given sorting criterion (often: minimal cardinality or maximal probability); A is *focused best-first* iff it is best-first only for a particular sorting criterion (often: minimal cardinality); A is *only-best* iff it computes only the best diagnosis (if its type is single-solution, cf. Bullet 4) / diagnoses (if its type is multiple-solution, cf. Bullet 4) wrt. a particular property (often: minimum-cardinality); A is *best-subset-no-order* iff it computes a subset of all diagnoses including exactly the best diagnoses wrt. a particular property (often: minimal cardinality), but the diagnoses are not computed or output in best-first order; A is *any-first* iff it does not satisfy any of the above conditions and cannot guarantee any particular output order of diagnoses; if A is any-first, but uses heuristic techniques to approximate a particular order of the computed diagnoses, it is *heuristic best-first*.
Relevance: The best-first property is useful, e.g., if (one expects) there is a large number of diagnoses and the actual diagnosis is likely among the best diagnoses (e.g., when all components fail with an equal and small likelihood [13]), if focusing techniques are adopted where only the best subset of all diagnoses is further considered [13], if informative samples for sequential diagnosis should be computed [35], or if users intend to monitor the best diagnoses throughout the debugging process [23].
Impact: Forgoing the best-first requirement usually leads to a higher efficiency of diagnosis computation, as any-first algorithms can use more efficient (e.g., depth-first [26] instead of breadth-first [12] or uniform-cost [19]) diagnosis search techniques. Also, generally best-first methods might be significantly more expensive than related focused best-first ones (cf., e.g., [23]).
Relationship: To the best of our knowledge, all generally best-first algorithms are conflict-dependent (cf. Bullet 5), i.e., rely on a systematic search based on conflicts (cf. Tab. 1). Moreover, compilation-based approaches (cf. Bullet 5) are usually only-best techniques wrt. minimum-cardinality diagnoses (cf. Tab. 1).

4. **Type of Output:**
Definition: A is called *multiple-solution* iff it can compute a set of two or more diagnoses per call; otherwise, if A returns at most one diagnosis, it is called *single-solution*.
Relevance: The optimal algorithm choice wrt. this feature is trivial and depends on whether one or multiple diagnoses are required in a scenario. Note, most algorithms considered in Tab. 1 can output multiple solutions. Clearly, any such algorithm can also be employed if only a single solution is desired.
Impact: Single-solution techniques can be highly performant as they may use optimizations that harm completeness by manipulating the set of all solutions for the benefit of computational efficiency [30]. To allow for some degree of control over their performance, multiple-solution approaches are sometimes also configurable, e.g., to compute a number of exactly k solutions, to stop after some timeout occurs, to prune a specified part of the search space, or to stop after a predefined number of search iterations has been performed [23; 36; 38; 16; 17].
Relationship: Single-solution methods are usually one-complete (cf. Bullet 2) and only-best (cf. Bullet 3), see Tab. 1. Simply put, when focusing on only one solution, approaches normally aim at finding the *best* diagnosis wrt. some property among *all* minimal diagnoses.

5. **Conflict Dependency:**
Definition: A is *conflict-dependent* iff it requires the computation of (minimal) conflicts; otherwise, i.e., if A works without taking information about conflicts into account, it is *direct*; a direct algorithm which translates the DPI into a target language and performs diagnosis computation based on this alternative problem representation is called *compilation-based*.
Relevance: If conflict computation or theorem proving is very expensive in a diagnostic scenario (cf., e.g., [9]), then direct algorithms are preferable, or even the only feasible approach. If a systematic exploration of diagnoses (allowing, e.g., inferences that all diagnoses with a specific property, say minimum-cardinality, are already computed) is desired [13] or the general best-first property (see bullet 3) is relevant [23], or the used method should be optimized for certain sequential diagnosis problems [24], then a conflict-dependent approach might be the only viable choice. When adopting a conflict-dependent method, it is important to note that an adequate conflict generation approach—which is sound and complete wrt. the computation of (minimal) conflicts as well as applicable to and performant for the DPI at hand—needs to be combined with the diagnosis algorithm. Choosing such an approach might not be an easy task for non-expert users.
Impact: Compilation-based techniques often allow to answer important diagnostic queries (such as minimum-cardinality diagnosis computation) in polynomial time in the size of the compilation (which however might be of exponential size). Hence, these methods might be the best choice given that the DPI at hand is amenable to a compact compiled representation. Direct techniques, some of which (e.g., [25;

26]) are based on the Duality Property (cf. Sec. 2), sometimes allow to escape computational bottlenecks concerning memory consumption [26] or time [16] by forgoing a *systematic* enumeration of the diagnoses. Most of the algorithms in the literature appear to be conflict-dependent (cf. Tab. 1), and most (but not all, e.g., [27]) of them are based on the Hitting-set Property (cf. Sec. 2); hence, there is a great selection of such methods, which cover numerous different combinations of other features, so that there will be a reasonable choice among them for many diagnosis tasks and applications.
Relationship: Considering the literature, it appears that conflict-dependency implies (if judgeable) the general applicability of an algorithm (cf. Bullet 9 and Tab. 1). The reason for this is that, for diagnosis computation, the set of minimal conflicts is representative of a DPI (cf. [28, Theorem 1]), and thus can be seen as a kind of general abstraction from the DPI, which can be applied to any DPI. Hence, algorithms relying on this abstraction (usually) do not make assumptions about system specifics.

6. **Way of Conflict Computation:**
Definition: (Prerequisite: A is conflict-dependent, cf. Bullet 5) A is *preliminary* iff it requires (a non-empty, non-singleton subset of) all minimal conflicts to be given as an input, or if it computes (a non-empty, non-singleton subset of) all minimal conflicts preliminarily, before the diagnosis computation starts; otherwise, if conflicts are computed on-demand in the course of diagnosis computation, A is *on-the-fly*.
Relevance: If the prior generation of all minimal conflicts is feasible or even practical in a diagnosis scenario, there are highly efficient preliminary techniques available for diagnosis computation (cf., e.g., [29; 30]). These preliminary techniques can also benefit from insights of a significant body of research regarding the minimal hitting set problem (cf. [31] for an overview). On-the-fly algorithms, on the other hand, are often still efficiently applicable even if preliminary conflict generation is infeasible. That said, it might in certain diagnostic use cases not be necessary to explicitly derive all (minimal) conflicts, e.g., in sequential diagnosis scenarios [13; 32] where only a subset of diagnoses is required per iteration. Some preliminary techniques (e.g., [7]) can be modified to act on-the-fly, but not all of them (e.g., ones that exploit the structure in the collection of minimal conflicts [33]). Any on-the-fly algorithm can be modified to be preliminary in a straightforward way (by pre-computing the collection of minimal conflicts and by choosing appropriate conflicts from this collection on-the-fly).
Impact: Forgoing the preliminary computation of the (full) set of minimal conflicts and intermixing conflict generation with diagnosis computation can allow to escape a combinatorial explosion and thus enhance the performance of diagnosis methods [13].
Relationship: Usually, preliminary algorithms do not incorporate mechanisms for generating minimal conflicts, but assume them to be given from the outset (e.g., [34; 33; 30]). For such methods, we cannot assess the features general applicability, black-box reasoning, and logics-agnosticism (cf. Bullets 9, 10 and 11) because these methods do not directly use the DPI, but require some "external" technique to provide the required collection of conflicts, where the three said features above depend on the adopted conflict generation technique.

7. **Focus on Sequential Diagnosis:**
Definition: A is *sequential* iff it provides mechanisms to address the sequential diagnosis problem (cf. Problem 2), e.g., in terms of measurement proposal techniques or system knowledge update procedures after measurement actions; otherwise, A is *one-shot*.
Relevance: To solve a sequential diagnosis problem, algorithms devised specifically for this purpose will often be more practical than iteratively re-invoking a one-shot algorithm for the various DPIs (successively extended by new measurements, cf. Sec. 2) during a sequential diagnosis session (cf., e.g., [36; 37]). Apart from that, the former techniques will often be directly applicable to a sequential diagnosis task, whereas a user might need to adapt the implementation of a one-shot algorithm to make it ready for sequential diagnosis. On the other hand, if sequential diagnosis is not the task in a diagnostic scenario, then a user is generally better off (wrt. efficiency, implementation complexity, etc.) when using one of the often less sophisticated (cf., e.g., [24]) one-shot techniques.
Impact: Relying on sequential techniques will usually boost the performance of diagnosis computation in a sequential setting, but will generally also tend to worsen the performance in non-sequential settings.
Relationship: Sequential techniques are usually sound, complete and stateful (cf. Bullets 1, 2, and 8, and Tab. 1) where the former two properties can be useful for diagnostic decision-making (e.g., measurement proposal, stop criteria) and the latter can improve the time performance of an algorithm (cf. [24; 7]).

8. **Maintenance of State:**
Definition: A is *stateful* iff it can maintain its state when used throughout a sequential diagnosis process (e.g., by storing or reusing data structures, intermediate values, etc.); otherwise, A is *stateless*.
Relevance: Since this feature describes the internal workings of an algorithm, users might basically be indifferent whether the used diagnosis method is stateful or stateless. However, requirements wrt. the algorithm's performance may (ceteris paribus) have a bearing on the proper choice between stateful and stateless algorithms (see below).
Impact: When memory is the more critical resource, e.g., on small or mobile devices, stateless algorithms may be a way to trade more time for less space, whereas, when time is the more critical resource, stateful algorithms may be preferable [24; 38].
Relationship: Stateless algorithms are usually (but not always, cf. [26]) one-shot (cf. Bullet 7), and algorithms that can be used in a stateful way are normally (but not always, cf. [39]) sequential (cf. Bullet 7). See Tab. 1.

9. **General Applicability:**
Definition: A is *generally applicable* iff it can be used for any diagnosis problem expressible by means of Reiter's theory [12], i.e., for any DPI as specified in Sec. 2; otherwise, e.g., if A makes certain assumptions about (e.g., the structure or some properties of)

the tackled DPI, it is *problem-dependent*.

Relevance: The appropriate choice of diagnosis algorithm depends on its application area and scope. E.g., if only certain system types (such as circuits) are addressed, then a problem-dependent algorithm that considers and leverages the peculiarities of this system type will be the proper and often much more performant approach (cf. [40; 37; 39]). If, on the other hand, a diagnosis system's intended use is for frequently changing application domains (e.g., in the Semantic Web context, where a multitude of different domains are modeled in terms of ontologies with highly heterogeneous content, structure, expressiveness, reasoning complexity and used logical languages [4; 19; 23]), problem-dependent techniques might not be eligible and generally applicable ones allow to deal with various diagnosis problems without modifying the diagnosis system.

Impact: If general applicability is required, the price to pay for this is the use of general-purpose diagnostic techniques which naturally cannot match up in terms of performance to approaches geared to optimizing diagnostic efficiency for specific problem cases.

Relationship: General applicability implies logics-agnosticism (cf. Bullet 11) since an algorithm incapable of dealing with some (monotonic) logical language is, by definition, not generally applicable. However, there might be logics-agnostic techniques which exploit structural properties of a particular system type (regardless of how it is modeled), which are thus not generally applicable. Moreover, most (but not all, cf. [7]) of the generally applicable methods are black-box (wrt. reasoning), i.e., can use an arbitrary (sound and complete) inference mechanism (cf. Bullet 10).

10. **Black-Box Reasoning:**

Definition: A is *black-box* (wrt. reasoning) iff it uses a reasoner as a black-box oracle (for consistency or model checking) and can use an arbitrary (sound and complete) reasoner for the logical language used to express the DPI; otherwise, if A requires additional computations or mechanisms (e.g., operations pertinent to a specific problem representation [41; 42] or bookkeeping techniques [7]) from a reasoner beyond the main reasoning result, it is *reasoner-dependent*.[2]

Relevance: If the logic used to model the diagnosed system is stable in an application area, then reasoner-dependent approaches might be the better choice as they might be advantageous in terms of diagnostic efficiency (given a suitable "glass-box" reasoner for the respective logic). E.g., when DPIs expressible by means of propositional logic are the target use case of a diagnosis system, then a reasoner-dependent algorithm based on propositional logic might be preferable to a black-box one with otherwise equal features. If, on the other hand, different formalisms might be used to describe the faulty system [23], black-box techniques can be more expedient. They can always simply use the best reasoner for the particular problem at hand without needing to incorporate any modifications into the reasoner (or the diagnosis algorithm)—unlike reasoner-dependent approaches, which require the incorporation of the necessary additional mechanisms into any adopted reasoner. And, performances of various reasoners might differ substantially [46]. As a rule of thumb, if the performance for one *fixed* system description language should be maximized, then reasoner-dependent approaches tend to be more favorable, whereas black-box methods tend to be preferable if the performance over *variable* modeling languages should be optimized.

Impact: Reasoner-dependent techniques can lead to an improved time performance [44; 45], e.g., when reasoners extract conflicts as a byproduct of consistency checks [44; 45] or store supporting environments[3] for derived entailments [7], but might also incur memory overheads [47]. Advantages of black-box methods are, e.g., their robustness (no sophisticated, and potentially error-prone, modifications of complex reasoning algorithms), their simplicity (internals of reasoner irrelevant), their flexibility (e.g., black-box methods can use a portfolio reasoning approach by switching to the most efficient reasoner in a simple plug-in fashion depending on the language used to describe the diagnosed system [48]), and their up-to-dateness (black-box methods can directly benefit from advances in the general research on automated reasoning).

Relationship: There are no general implications on other features resulting from the presence or absence of the black-box property; however, it is often (but not always) the case that black-box techniques are also generally applicable, logics-agnostic, sound, complete and multiple-solution (cf. Bullets 1, 2, 9, 11 and 4).

11. **Logics-Agnosticism:**

Definition: A is *logics-agnostic* iff it can deal with DPIs expressed by arbitrary (monotonic) logics; otherwise, A is *logics-dependent*.

Relevance: If a diagnosis approach is intended to be used with only one *fixed* system description language, then a user should choose the method with (expected/reported) best performance for the faced diagnostic task, regardless of whether it is logics-agnostic or -dependent. Logics-dependent approaches, however, can offer very attractive features, e.g., compilation-based approaches (e.g., [39; 41; 42]) can often compute diagnoses in polynomial time once the DPI has been compiled into a target representation; however, they are restricted to propositional logic system descriptions. If a diagnosis approach needs to deal with *diverse* system modeling languages, the adoption of a logics-agnostic method might be the only choice.

Impact: As our literature study suggests, logics-dependent approaches are usually particularly attractive in terms of performance. The reason for this is that these methods often use sophisticated (e.g., representation, optimization or reasoning) techniques specific to one logic, which is propositional (Horn) logic for all logics-dependent approaches considered in Tab. 1.

Relationship: The property of logics-agnosticism appears to come along with the black-box property (cf. Bullet 10) in most (but not all, cf. [7]) cases. Furthermore, logics-agnostic techniques are often generally applicable (cf. Bullet 9).

[2] Methods we call reasoner-dependent are sometimes also referred to as *glass-box* techniques [43; 44; 45].

[3] Roughly: sets of logical sentences sufficient for the entailment to hold. Environments are also termed *justifications* [44].

12. **Space Efficiency:**
    Definition: A is *space-efficient* iff its worst-case memory complexity is polynomial in the input size; otherwise, A is *space-inefficient*.
    Relevance: If time is the resource to be minimized and the diagnostic task is expected to manage with the available memory, then space-inefficient methods can be the better choice, due to their generally lower time complexity. If, on the other hand, the available memory capacity of a device is low (such as with IoT or smaller mobile devices) or the diagnostic task is expected to be memory-intensive (e.g., if diagnosis cardinality is likely to be high [26]), then space-efficient algorithms might be the only viable approach (as memory consumption increases exponentially with the size of diagnoses for many space-inefficient techniques). Note, only few space-efficient diagnosis computation strategies have been proposed in literature, thus there is not plenty of choice for the user, which is why a trade-off between space-efficiency and other properties might be necessary. Only recently, a space-efficient algorithm exhibiting all desirable properties wrt. the features discussed here, along with a reasonable time performance, has been suggested [23].
    Impact: Space-efficiency is often bought for a higher (empirical or theoretical) time complexity [23], or for a dropping of other desirable properties such as best-firstness [26; 25] or completeness [17].
    Relationship: With the exception of the algorithm proposed in [23], space-efficiency appears to be achievable only for algorithms that do not guarantee a best-first diagnosis computation (cf. Bullet 3), or that require a preliminary computation of conflicts (cf. Bullet 6) and whose complexity thus does not take into account the conflict generation phase.

Remarks:

- We do not propose a feature concerning the time complexity. This is due to well-known complexity results [49; 50], which imply (unless P = NP) that there cannot be an algorithm that computes at least two minimal diagnoses and generally finishes in polynomial time; this holds even if reasoning (consistency checking) is in P, which however is already NP-complete if (only) propositional logic models are used (let alone more expressive logics such as Description Logics [51]).
- The list of proposed features is certainly not an exhaustive account of all possible properties diagnosis computation algorithms might have. There are further conceivable aspects from the *(a) theoretical viewpoint*, such as whether an algorithm uses abstractions or alterations of a DPI, or whether computations are executed in a distributed or centralized way (cf. [24, Sec. 4]), *(b) empirical viewpoint*, such as whether an algorithm was experimentally evaluated, which dataset from which domain was used to evaluate an algorithm, or which other methods an algorithm was compared against, *(c) presentation viewpoint*, such as whether formal proofs for algorithm properties are given, or from the *(d) pragmatic viewpoint*, such as whether there are freely accessible implementations of or tools relying on an algorithm. Exploring other features like these is on our future work agenda.

4 Classification of Existing Works

Tab. 1 gives a classification of several existing works based on their characteristics wrt. the features suggested in Sec. 3.
Remarks:

- The table can be read row-wise to inspect the features of the diagnostic techniques, and column-wise to find methods with certain characteristics wrt. the features.
- We assessed the algorithms enumerated in the table as they are described in the respective cited work, without assuming any modifications or extensions.
- The list of algorithms studied in the table raises no claim to completeness. Rather, the idea is to illustrate the use(fulness) of the presented features for algorithm assessment and comparison by presenting the properties of *some* important methods in literature. We plan to analyze further ones as part of our future work.

5 Conclusions

This work proposes a taxonomy for diagnosis computation algorithms, with the intention of helping researchers and practitioners in assessing, comparing, and selecting diagnostic techniques for their tasks and purposes. More specifically, we present a set of 12 crucial features of diagnosis techniques and classify some important existing methods based on these. In our study of the works in the literature, we observed that, for some algorithms, it was relatively hard to determine their properties regarding the proposed taxonomy since various algorithmic aspects are often left implicit or not addressed at all. Moreover, even if properties are discussed, not all works provide formal proofs of these (which is certainly not least because of strict space restrictions at many publication venues). Hence, we encourage authors (whenever possible) to *explicitly* discuss the material properties of their proposed diagnostic approaches, for which we hope the suggested taxonomy will constitute a useful basis and guideline. Making algorithm characteristics explicit, clear and accessible will certainly help other researchers put novel works appropriately into the context of existing works, and sticking to shared assessment and categorization criteria while doing so can have several advantages. Examples are *(a)* fair empirical evaluations that contrast methods which are actually comparable (e.g., comparing an incomplete method against a complete one wrt. performance might under circumstances be pointless, as the methods simply accomplish different things and have different use cases), *(b)* the potential inspiration for and detection of open research questions (e.g., find an algorithm which has a particular subset of the features, which are not met by any of the existing algorithms), *(c)* an easier, faster and more informed finding of a suitable algorithm for a particular purpose (e.g., is there a space-efficient method for a mobile device that is at the same time sound, complete and best-first?), *(d)* the better understanding of the evolution and reality in research and practice (e.g., that certain feature combinations are the reason why some techniques are not used in some application domains while they are state-of-the-art in others), or *(e)* the realization that basically all algorithms have their right to exist, as they cover a wide variety of feature combinations and thus address a broad range of diagnostic problem scenarios, and that algorithms "superseding" others due to performance improvements most often achieve this at the cost of losing some desirable properties.

References

[1] J Hunt. Model-based software diagnosis. *Applied Artificial Intelligence*, 12(4):289–308, 1998.

[2] T Eiter, M Fink, and D Stepanova. Data repair of inconsistent dl-programs. *IJCAI*.

[3] A Felfernig, G Friedrich, and E Teppan. Automated Debugging and Repair of Utility Constraints in Recommender Knowledge Bases. In *DX*, 2007.

[4] K Shchekotykhin, G Friedrich, P Fleiss, and P Rodler. Interactive Ontology Debugging: Two Query Strategies for Efficient Fault Localization. *JWS*, 12-13:88–103, 2012.

[5] P Rodler, K Shchekotykhin, P Fleiss, and G Friedrich. RIO: Minimizing User Interaction in Ontology Debugging. In *RR*, 2013.

[6] G Friedrich, M Stumptner, and F Wotawa. Model-based diagnosis of hardware designs. *AIJ*, 111(1-2):3–39, 1999.

[7] J de Kleer and B Williams. Diagnosing multiple faults. *AIJ*, 32(1):97–130, 1987.

[8] S Zaman, G Steinbauer, J Maurer, P Lepej, and S Uran. An integrated model-based diagnosis and repair architecture for ROS-based robot systems. In *ICRA*, 2013.

[9] P Rodler, E Teppan, and D Jannach. Randomized Problem-Relaxation Solving for Over-Constrained Schedules. In *KR*, 2021.

[10] D Gorinevsky, K Dittmar, D Mylaraswamy, and E Nwadiogbu. Model-based diagnostics for an aircraft auxiliary power unit. In *CCA*, 2002.

[11] M Sachenbacher, A Malik, and P Struss. From electrics to emissions: experiences in applying model-based diagnosis to real problems in real cars. In *DX*, 1998.

[12] R Reiter. A Theory of Diagnosis from First Principles. *AIJ*, 32(1):57–95, 1987.

[13] J de Kleer. Focusing on Probable Diagnoses. In *AAAI*, 1991.

[14] J de Kleer, A Mackworth, and R Reiter. Characterizing diagnoses and systems. *AIJ*, 56, 1992.

[15] J de Kleer and O Raiman. How to diagnose well with very little information. In *DX*, 1993.

[16] A Feldman, G Provan, and A van Gemund. Computing Minimal Diagnoses by Greedy Stochastic Search. In *AAAI*, 2008.

[17] R Abreu and A van Gemund. A Low-Cost Approximate Minimal Hitting Set Algorithm and its Application to Model-Based Diagnosis. In *SARA*, 2009.

[18] D Heckerman, J Breese, and K Rommelse. Decision-theoretic troubleshooting. *Comm. of the ACM*, 38(3):49–57, 1995.

[19] P Rodler. *Interactive Debugging of Knowledge Bases*. PhD Thesis, Univ. of Klagenfurt, 2015.

[20] A Blumer, A Ehrenfeucht, D Haussler, and M Warmuth. Occam's razor. *Inf. Process. Lett.*, 24(6):377–380, 1987.

[21] L Li and J Yunfei. Computing minimal hitting sets with genetic algorithm. In *DX*, 2002.

[22] J Yunfei and L Li. Computing the minimal hitting sets with binary HS-tree. *J. Softw.*, 13(12):2267–2274, 2002.

[23] P Rodler. Memory-limited model-based diagnosis. *AIJ*, 305:103681, 2022.

[24] P Rodler. DynamicHS: Streamlining Reiter's Hitting-Set Tree for Sequential Diagnosis. *arXiv:2012.11078*, 2020.

[25] A Felfernig, M Schubert, and C Zehentner. An efficient diagnosis algorithm for inconsistent constraint sets. *AI-EDAM*, 26(1):53–62, 2011.

[26] K Shchekotykhin, G Friedrich, P Rodler, and P Fleiss. Sequential diagnosis of high cardinality faults in knowledge-bases by direct diagnosis generation. In *ECAI*, 2014.

[27] B Williams and R Ragno. Conflict-directed A* and its role in model-based embedded systems. *Discrete Appl. Math.*, 155(12):1562–1595, 2007.

[28] J de Kleer. Mininimum cardinality candidate generation. In *DX*, 2009.

[29] L Shi and X Cai. An Exact Fast Algorithm for Minimum Hitting Set. In *CSO*, 2010.

[30] J de Kleer. Hitting set algorithms for model-based diagnosis. In *DX*, 2011.

[31] A Gainer-Dewar and P Vera-Licona. The minimal hitting set generation problem: algorithms and computation. *SIDMA*, 31(1):63–100, 2017.

[32] P Rodler. On Active Learning Strategies for Sequential Diagnosis. In *DX*, 2017.

[33] X Zhao and D Ouyang. Deriving all minimal hitting sets based on join relation. *IEEE Trans. Syst. Man Cybern.: Syst.*, 45(7):1063–1076, 2015.

[34] L Li and J Yunfei. The computation of hitting sets: Review and new algorithms. *Inf. Process. Lett.*, 86(4):177–184, 2003.

[35] P Rodler. Random vs. Best-First: Impact of Sampling Strategies on Decision Making in Model-Based Diagnosis. In *AAAI*, 2022.

[36] P Rodler. Reuse, Reduce and Recycle: Optimizing Reiter's HS-Tree for Sequential Diagnosis. In *ECAI*, 2020.

[37] S Siddiqi and J Huang. Sequential diagnosis by abstraction. *JAIR*, 41:329–365, 2011.

[38] P Rodler and M Herold. StaticHS: A Variant of Reiter's Hitting Set Tree for Efficient Sequential Diagnosis. In *SoCS*, 2018.

[39] A Metodi, R Stern, M Kalech, and M Codish. A novel SAT-based approach to model based diagnosis. *JAIR*, 51:377–411, 2014.

[40] A Feldman and A van Gemund. A two-step hierarchical algorithm for model-based diagnosis. In *AAAI*, 2006.

[41] P Torasso and G Torta. Model-based diagnosis through OBDD compilation: A complexity analysis. In *Reasoning, Action and Interaction in AI Theories and Systems*, 2006.

[42] A Darwiche. Decomposable negation normal form. *JACM*, 48(4):608–647, 2001.

[43] B Parsia, E Sirin, and A Kalyanpur. Debugging OWL ontologies. In *WWW*, 2005.

[44] M Horridge. *Justification based Explanation in Ontologies*. PhD Thesis, Univ. of Manchester, 2011.

[45] A Kalyanpur. *Debugging and Repair of OWL Ontologies*. PhD Thesis, Univ. of Maryland, College Park, 2006.

[46] R Gonçalves, S Bail, E Jiménez-Ruiz, N Matentzoglu, B Parsia, B Glimm, and Y Kazakov. OWL reasoner evaluation (ORE) workshop 2013 results. In *ORE*, 2013.

[47] A Kalyanpur, B Parsia, E Sirin, and J Hendler. Debugging Unsatisfiable Classes in OWL Ontologies. *JWS*, 3(4):268–293, 2005.

[48] A Romero, B Cuenca Grau, and I Horrocks. MORe: Modular combination of OWL reasoners for ontology classification. In *ISWC*, 2012.

[49] T Eiter and G Gottlob. The complexity of logic-based abduction. *JACM*, 42(1):3–42, 1995.

[50] T Bylander, D Allemang, M Tanner, and J Josephson. The computational complexity of abduction. *AIJ*, 49:25–60, 1991.

[51] F Baader, D Calvanese, D McGuinness, D Nardi, and PF Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2007.

[52] R Greiner, B Smith, and R Wilkerson. A correction to the algorithm in Reiter's theory of diagnosis. *AIJ*, 41(1):79–88, 1989.

[53] A Hou. A theory of measurement in diagnosis from first principles. *AIJ*, 65(2):281–328, 1994.

[54] F Wotawa. A variant of Reiter's hitting-set algorithm. *Inf. Process. Lett.*, 79(1):45–51, 2001.

[55] Z Xiangfu and O Dantong. A method of combining SE-tree to compute all minimal hitting sets. *Prog. Nat. Sci.*, 16(2):169–174, 2006.

[56] R Stern, M Kalech, A Feldman, and GM Provan. Exploring the Duality in Conflict-Directed Model-Based Diagnosis. In *AAAI*, 2012.

[57] I Pill and T Quaritsch. Optimizations for the Boolean approach to computing minimal hitting sets. In *ECAI*, 2012.

[58] D Jannach, T Schmitz, and K Shchekotykhin. Parallel model-based diagnosis on multi-core computers. *JAIR*, 55:835–887, 2016.

[59] A Darwiche. New advances in compiling CNF to decomposable negation normal form. In *ECAI*, 2004.

Technique			Features											
			Output Qualities				Way of Computation		Seq. Diag. Context		Application Context			Performance
Name	Year	Work	SOUND	COMPL	BEST-F	MULT-SOL	CONF-DEP	O-T-FLY	SEQ	STATE	GEN-APPL	BL-BOX-REAS	ANY-LOGIC	POLY-SPACE
GDE	1987	[7]	✓	✓(all)	✓(gen)	✓	✓	×	✓	✓	✓	×(bk)	✓	×
HS-Tree	1987	[12]	✓	✓(all)	✓(foc=mc)	✓	✓	✓	×	×	✓	✓	✓	×
HS-DAG	1989	[52]	✓	✓(all)	✓(gen)	✓	✓	✓	×	×	✓	✓	✓	×
DIAGNOSE	1994	[53]	✓	✓(all)	×	✓	✓	×	✓	✓	✓	✓	✓	×
HST	2001	[54]	✓	✓(all)	✓(foc=mc)	✓	✓	✓	×	×	✓	✓	✓	×
DNNF	2001	[42]	✓	✓(p=mc)	✓(only=mc)	✓	×(cp-b=DNNF)	na	×	?	×	×	×(PL)	×
Genetic Alg.	2002	[21]	×	×	×	✓	✓	×	×	×	na	na	na	?
BHS-Tree	2003	[34]	✓	✓(all)	×	✓	✓	×	×	×	na	na	na	×
Bool. Alg.	2003	[34]	✓	✓(all)	×	✓	✓	×	×	×	na	na	na	×
HSSE-Tree	2006	[55]	✓	✓(all)	✓(foc=mc)	✓	✓	×	×	×	na	na	na	×
HA*	2006	[40]	✓	✓(one=mc)	✓(only=mc)	×	×(cp-b=h-DNF)	na	×	?	×(circ)	✓	×(PL)	×
OBDD	2006	[41]	∼(1)	✓(all)	✓(only=mc)	✓	×(cp-b=OBDD)	na	✓	✓	×	×	×(PL/HL)	×
CDA*	2007	[27]	✓	×	✓(gen)	✓	✓	✓	×	?	✓	✓	✓	×
SAFARI	2008	[16]	∼(2)	×	×	✓	×(dir)	na	×	×	✓	✓	✓	×
STACCATO	2009	[17]	?	∼(3)	×(heur)	✓	✓	×	×	×	na	na	na	✓
NGDE	2009	[28]	✓	✓(p=mc)	✓(only=mc)	✓	✓	✓	×	?	✓	×(bk)	✓	?(4)
Recurs. MHS	2010	[29]	✓	✓(one=mc)	✓(only=mc)	×	✓	×	×	na	na	na	na	∼(5)
SDA	2011	[37]	✓	✓(one=SD-sol)	✓(only=SD-sol)	×	×(cp-b={BN, d-DNNF})	na	✓	✓	×(circ)	×	×(PL)	×(6)
cminc	2011	[30]	✓	✓(one=mc)	✓(only=mc)	×	✓	×	×	na	na	na	na	?(7)
FastDiag	2011	[25]	✓	✓(all)	×	✓	×(dir)	na	×	×	✓	✓	✓	✓
SDE	2012	[56]	✓	✓(all)	✓(foc=mc)	✓	✓	✓	×	×	✓	✓	✓	×
Improved Bool. Alg.(*)	2012	[57]	✓	✓(all)	✓(bsno)	✓	✓	×	×	na	na	na	na	×
Inv-HS-Tree	2014	[26]	✓	✓(all)	×	✓	×(dir)	na	✓	×	✓	✓	✓	✓
SATbD	2014	[39]	✓	✓(p=mc)	✓(only=mc)	✓	×(cp-b=SAT)	na	×	✓	×(circ)	✓	×(PL)	?
Increm-distrib-MHS(*)	2015	[33]	✓	✓(all)	×	✓	✓	×	✓	✓	na	na	na	×
Unif-cost HS-Tree(*)	2015	[19]	✓	✓(all)	✓(gen)	✓	✓	✓	×	×	✓	✓	✓	×
Parallel HS-Tree	2016	[58]	✓	✓(all)	✓(foc=mc)	✓	✓	✓	×	×	✓	✓	✓	×
StaticHS	2018	[38]	✓	✓(all)	✓(gen)	✓	✓	✓	✓	✓	✓	✓	✓	×
DynamicHS	2020	[36]	✓	✓(all)	✓(gen)	✓	✓	✓	✓	✓	✓	✓	✓	×
RBF-HS	2022	[23]	✓	✓(all)	✓(gen)	✓	✓	✓	×	×	✓	✓	✓	✓
HBF-HS	2022	[23]	✓	✓(all)	✓(gen)	✓	✓	✓	×	×	✓	✓	✓	×
Heuristic Inv-HS-Tree(*)	2022	[35]	✓	✓(all)	×(heur)	✓	×(dir)	na	×	×	✓	✓	✓	✓

Table 1: Classification of some existing diagnosis computation algorithms based on their characteristics wrt. the features proposed in Sec. 3. Table rows are sorted by the year of publication of the algorithms. Features (table columns) are thematically grouped as described in Sec. 3. *(Column meanings:)* SOUND...is the algorithm sound? (Bullet 1); COMPL...is the algorithm complete? (Bullet 2); BEST-F...is the algorithm best-first? (Bullet 3); MULT-SOL...is the algorithm multiple-solution? (Bullet 4); CONF-DEP...is the algorithm conflict-dependent? (Bullet 5); O-T-FLY...does the algorithm (if applicable) compute conflicts on-the-fly? (Bullet 6); SEQ...is the algorithm sequential? (Bullet 7); STATE...is the algorithm stateful? (Bullet 8); GEN-APPL...is the algorithm generally applicable? (Bullet 9); BL-BOX-REAS...is the algorithm black-box wrt. reasoning? (Bullet 10); ANY-LOGIC...is the algorithm logics-agnostic? (Bullet 11); POLY-SPACE...is the algorithm space-efficient? (Bullet 12). *(Symbol meanings:)* ✓...yes; ∼...under certain circumstances; ×...no; ✓(all)...all-complete; ✓(p=X)...property-complete (wrt. property X); ✓(one=X)...one-complete (wrt. property X); ✓(gen)...generally best-first; ✓(foc=X)...focused best-first (wrt. property X); ✓(only=X)...only-best (wrt. property X); ✓(bsno=X)...best-subset-no-order (wrt. property X); ×(heur)...heuristic best-first; ×(cp-b=X)...compilation-based (using target language X); ×(dir)...direct; na...not applicable; ×(circ)...specific to circuit diagnosis problems; ×(bk)...uses bookkeeping mechanism for reasoning (ATMS for [7], HTMS for [30]); ×(PL)...specific to propositional logic; ×(PL/HL)...specific to propositional logic / Horn logic; ?...unknown. *(Table notes:)* (*)...algorithm name given in the table not used in the original work; (1)...sound wrt. all diagnoses, but not wrt. all *minimal* diagnoses; (2)...unsound in most efficient configuration; can be parameterized to be sound; (3)...incomplete in most efficient configuration; can be parameterized to be complete; (4)...the diagnosis search is depth-first, i.e., requires linear space, but unclear if the used HTMS is polynomial space; (5)...polynomial-space minimal hitting set computation, but assumes a given (i.e., preliminarily computed) set of conflicts; (6)...d-DNNF is less succinct (i.e., larger in general) than DNNF [59, Sec. 2], and compilation to DNNF may result in exponential-size compilations [42]; (7)...uses depth-first search, which requires linear space, but unclear if FullReduce() function [30], which has to consider all conflicts at once, is polynomial space.